DART: A Simulation Code for Charged Particle Beams

R.C. White

W.L. Barr

R.W. Moir

UCID--21330

May 16, 1988

388 011757

Abstract

This paper presents a recently modified version of the 2-D DART code designed to simulate the behavior of a beam of charged particles whose paths are affected by electric and magnetic fields. This code was originally used to design laboratory-scale and full-scale beam direct converters. Since then, its utility has been expanded to allow more general applications. The simulation technique includes space charge, secondary electron effects, and neutral gas ionization. Calculations of electrode placement and energy conversion efficiency are described.

Basic operation procedures are given including sample input files and output.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

CONTENTS

1	Introduction	1
2	Running DART	2
3	Input Parameters	2
3.1	Parameters Contained in Namelist /nl1/	3
3.2	Parameters Contained in Namelist /nl2/	4
3.3	Parameters Contained in Namelist /nl3/	8
3.4	Parameters Contained in Namelist /nl4/	9
3.5	Parameters Contained in Namelist /nl5/	10
4	Input and Output Files	11
4.1	Input File DARTIN	11
4.2	Output File DARTOUT	12
5	What DART Does	12
5.1	Solving the Equation of Motion	15
5.1.1	Calculating Forces	15
5.2	Space Charge Calculations	15
5.3	Boundaries: Transparent, Fixed, Symmetry	16
5.4	Starting Beams at Various Angles	17
5.5	Determining End Points and Efficiencies	17
5.5.1	Conditions for Terminating Trajectory Calculations	17
5.5.2	Efficiency Calculations and Energy Conversion Check	17
5.6	Secondary Electrons	18
5.7	Cold Ions	18
6	Major DART Subroutines	21
6.1	Designating Electrodes	21
6.1.1	GETGRID	21
6.1.2	ELECTROD	21
6.2	POT1	21
6.3	POT	24
6.4	START	24
6.5	The ODE Solver	25
6.5.1	DRIVESA, STIFFA, COSETTA, and FUNCT	25
6.6	PHI and LAG2	25
6.7	PARPLT	25
6.8	SECPAR	25
6.9	CIPROD	26
6.10	QDENT	26
6.11	IMP	29
7	Modeling Electrons	29
8	A Beam with Space Charge—Test Case	30

9	The Child-Langmuir Law—Test Case	32
10	A Test Case for Cold Ion Calculations	34
11	Default Geometry	36
12	Summary	37

1 Introduction

In 1971, a simulation code for a direct energy converter for use in fusion reactors was written by B.C. Howard in collaboration with W.L. Barr and R.W. Moir at Lawrence Livermore National Laboratory [1]. The code was developed for use in designing a direct energy converter system. The code has recently been revised to give it a wider variety of applications by allowing arbitrary placement of electrodes. This document describes the new (1988) version of DART. Although further modifications may be made, the operation of the code and the majority of input parameters will be preserved so that the utility of this document will not be severely jeopardized.

The basic calculations the code performs are as follows:

1. Read initial conditions and electrode data from the file, DARTIN.
2. Display the data and prompt the user for changes.
3. Construct internal electrode information from data.
4. Solve Laplace's equation for the vacuum potential.
5. Initialize the primary ion trajectories for a beamlet.
6. Calculate electric and magnetic forces, solve the equation of motion, record charge density data, and plot trajectories of charged particles for this beamlet.
7. Determine trajectory end points (impact locations) and energy conversion efficiencies for this beamlet.
8. Start secondary electrons (if *ispar* is not zero) from each ion trajectory impact point, with an energy-dependent emission coefficient.
9. Plot secondary electron trajectories (if *ispar* is not zero), and summarize secondary electron data.
10. Repeat the above steps for each beamlet.
11. Start cold ion trajectories from selected mesh points, weighted with the sum of production rates gathered from primary ion trajectories.
12. Plot cold ion trajectories (if *icold* is not zero), and summarize cold ion data, including total efficiencies.
13. Use Poisson's equation to determine a new space charge potential table for further calculations.
14. Repeat steps 5 through 13 for *itmax* number of space charge iterations; then terminate the run.

Beams consist of superimposed components, or beamlets. Each beamlet has its own energy, starting position, width, current distribution, and orientation. Each beamlet is approximated by a finite set of trajectories and has its own starting parameters. The electric field results from a potential function whose values are interpolated from a two-dimensional table to give accurate results. Initially, vacuum potential is calculated. Then, space charge is included in a new potential calculation. For a specified number of trajectory iterations, charged particle trajectories and new

space charge are calculated. The number of trajectory iterations is determined by the input parameter called *itmar*. To continue with further space charge trajectory iterations after the run is complete, DART can be run again but with *isc* = 1 instead of *isc* = 0.

Each ion trajectory is followed until it terminates on either an electrode or wall, and the efficiency of energy conversion is calculated. This may provide useful information for an analysis of individual beam components. After all the beam components have been run, the total efficiency of the energy conversion is computed.

2 Running DART

The software for DART is DARTL, available from the National Energy Software Center (for more information, see Acknowledgments). Compilation and loading are necessary to obtain the executable version of DART. To compile and load the code on the Cray computer, use the CIVIC compiler and type:

$$\text{civic } i = \text{dartl} / t \ v$$

This step must be redone if any changes are made to the source, *dartl*.

We begin with a short description of the DART code, its input, output, and graphics.

The necessary libraries are loaded automatically. These consist of FORTLIB and GRAFL3. FORTLIB is the standard Fortran library on the Cray computers, while GRAFL3 is a collection of easy-to-use graphics modules based on the more sophisticated graphics library called GRAFLIB [2].

Running the code requires an input file called DARTIN. This file may be constructed or modified with your favorite editor. Editors supported on the Cray computers include TRIUGL and TEDI. (Examples of input files are provided in Sections 4, 8, 9, and 10.) Then simply type:

$$\text{dart} / t \ v$$

All units are SI, except that potentials are measured in kilovolts, energies in kilo-electron-volts, and gas pressure in Torr.

3 Input Parameters

All input parameters for this code are arranged in namelist blocks, allowing them to be grouped and displayed for user interaction with the DART code.

Initialization of these parameters is done with one input file, DARTIN. Variables not set in DARTIN will be equal to their default values, given in the sections which follow. Changes can be made to DART input parameters when the user is prompted by the DART code. These changes override the input file values for this run without changing DARTIN.

DARTIN contains namelist input with the variables listed in the sections which follow. A short description accompanies each variable along with its default value (shown in parentheses). The namelists are labeled by /n1/, /n2/, /n3/, /n4/, and /n5/. Changes to the first two namelists can only be made in DARTIN, whereas the last three are written to the terminal, where changes can be made to them. Details of how to use DARTIN and how to input from the terminal are described in Section 4.

These lists of input parameters, presented in the order in which they are accepted by the DART code, are intended for use as an input parameter directory.

Figure 1: Diagram showing grid spacing and location of various input parameters for the default test case (see Section 4.1).

3.1 Parameters Contained in Namelist /nl1/

imin: The minimum x -grid value. This variable determines the location of the left wall. Its dimensionless value represents grid-point units. *imin* must be greater than or equal to zero (default: 0). For an example of this and the following parameters, see Fig. 1.

imax: The maximum x -grid value. This variable determines the location of the right wall. Its dimensionless value represents grid-point units. *imax* must be less than or equal to 150 (default: 60).

kmin: The minimum z -grid value. This variable determines the location of the lower wall or symmetry plane. Its dimensionless value represents grid-point units. *kmin* must be greater than or equal to zero, and should usually be equal to zero (default: 0).

kmax: The maximum z -grid value. This variable determines the location of the upper wall. Its dimensionless value represents grid-point units. *kmax* must be less than or equal to 150 (default: 40).

xmax: Maximum value of x coordinate. This variable and *imax* determine the grid spacing in the x direction. *imax* is the maximum grid-point value, and *xmax* is the distance between this point and zero (default: 0.6).

zmax: Maximum value of z coordinate. This variable and *kmax* determine the grid spacing in the z direction. *kmax* is the maximum grid-point value, and *zmax* is the distance between this point and zero (default: 0.4).

vref: The potential value at which ions are started. *vref* is used in efficiency calculations. It is also used to determine which potential contours will be solid (greater than *vref*) and which will be dashed (less than *vref*) (default: -100.0).

isym*l*(*l*): *l* = 1,2,3,4 symmetry plane designator. *isym*(1) = 1 implies a symmetry plane at bottom: $z = 0$; *isym*(2) = 1 implies a symmetry plane at top: $z = z_{max}$; *isym*(3) = 1 implies a symmetry plane at left: $x = 0$; *isym*(4) = 1 implies a symmetry plane at right: $x = z_{max}$ (default values: *isym* = 1, 0, 0, 0).

3.2 Parameters Contained in Namelist /nl2/

eps1: Convergence parameter for over-relaxation. When a potential table is being calculated, consecutive iterations of the relaxation loop are compared. If the rms value of the difference between these successive potential values for all points involved is less than *eps1* times a measure of convergence, the process is terminated. To get more accurate results, we can decrease the value of *eps1*, but the process will take more time to converge (default: 0.005).

kpp1: The *z*-grid value at which to output potential values as a check on relaxation convergence. Values are written to the output file DARTOUT at regular intervals of the *x* coordinate. Its dimensionless value represents grid-point units (default: 2).

iterm: Maximum number of iterations to perform for potential relaxation. This is a limit to avoid spending too much time in the space charge calculations. Typically the actual number of iterations should be less. If not, *eps1* may be adjusted. The lists of sample potential values for each iteration given in DARTOUT are useful to check this calculation (default: 800).

ielmod: Switch for modeling electrons. *ielmod* = 0 indicates that no electrons are to be used. *ielmod* = 1 indicates that a Boltzmann distribution is to be used. *ielmod* = 2 clips the potential at a specified value. For details, see Section 7 (default: 1).

te: Electron temperature for use in Boltzmann distribution. See Section 7 (default: 0.01).

ine1: The *z*-grid coordinate where electron charge density begins, used in defining region one. In the default case, this region is near the suppressor. Its dimensionless value represents grid-point units (default: 0). Figures 2 and 3 show examples of this and the following parameters.

ipe1: The *z*-grid coordinate where electron charge density is cut off, thereby defining region one. In the default case, this region is near the suppressor. Its dimensionless value represents grid-point units (default: 12).

kne1: The *z*-grid coordinate where electron charge density begins, used to define region one. Its dimensionless value represents grid-point units (default: 0).

kpe1: The *z*-grid coordinate where electron charge density is cut off, thereby defining region one. Its dimensionless value represents grid-point units (default: 6).

ine2: The *z*-grid coordinate where electron charge density begins, used in defining region two. In the default case, this is the space not covered by region one. Its dimensionless value represents grid-point units. Note: Region one may be contained within region two, but region two is overridden by region one (default: 0).

Figure 2: Limits defining electron distribution parameters for the default test case. Region enclosed in dashed lines represents region one. Dotted lines surround regions for average ion charge density calculations. See Section 7.

Figure 3: Limits defining electron distribution parameters. Regions one and two are defined in the top diagram. In the bottom diagram, we define the areas where average ion charge densities are calculated. Horizontal shading indicates area for use in region one; vertical shading, for use in region two.

- ipe2:** The x -grid coordinate where electron charge density is cut off, thereby defining region two. Its dimensionless value represents grid-point units (default: 60).
- kne2:** The z -grid coordinate where electron charge density begins, used to define region two. Its dimensionless value represents grid-point units (default: 0).
- kpe2:** The z -grid coordinate where electron charge density is cut off to define region two. Its dimensionless value represents grid-point units (default: 40).
- il1:** Lower bound for the x -grid coordinate in area used to calculate average ion charge density in region one. This average ion charge density is used in the Boltzmann electron distribution. Its dimensionless value represents grid-point units (default: 0). For more on this calculation, see Section 7.
- iu1:** Upper bound for the x -grid coordinate in area used to calculate average ion charge density in region one. Its dimensionless value represents grid-point units (default: 2).
- kl1:** Lower bound for the z -grid coordinate in area used to calculate average ion charge density in region one. Its dimensionless value represents grid-point units (default: 0).
- ku1:** Upper bound for the z -grid coordinate in area used to calculate average ion charge density in region one. Its dimensionless value represents grid-point units (default: 4).
- il2:** Lower bound for the x -grid coordinate in area used to calculate average ion charge density for use in electron distribution modeling outside region one. Its dimensionless value represents grid-point units (default: 29).
- iu2:** Upper bound for the x -grid coordinate in area used to calculate average ion charge density for use in electron distribution modeling in region two. Its dimensionless value represents grid-point units (default: 60).
- kl2:** Lower bound for the z -grid coordinate in area used to calculate average ion charge density in region two. Its dimensionless value represents grid-point units (default: 0).
- ku2:** Upper bound for the z -grid coordinate in area used to calculate average ion charge density for use in electron distribution modeling in region two. Its dimensionless value represents grid-point units (default: 5).
- vg1:** Potential for referencing Boltzmann distribution or for cutoff in electron modeling in region one. In the default case, electrons are referenced to $vg1$ at the beam entry region. Region one is defined by specifying $ine1$, $ipe1$, $kne1$, and $kpe1$ (default: -100.0).
- vg2:** Potential for referencing Boltzmann distribution or for cutoff in electron modeling in region two. Region one has electrons modeled with respect to $vg1$, and region two has electrons modeled with respect to $vg2$ (default: 0).
- ichl:** Switch to run Child-Langmuir law test case. This switch should be set to one *only* when the sample input file shown in Section 9 is used. The generation of plots specific to that test case deems this necessary (default: 0).

Figure 4: Illustration of starting conditions for a beam.

3.3 Parameters Contained in Namelist /nl3/

iz: Charge state of primary ions. The charge on the ions is equal to iz times 1.6×10^{-19} C (default: 1).

na: Mass number of primary ions. The mass of the ions is equal to na times 1.67×10^{-27} kg. For a beam of deuterium ions, for example, set $na = 2$ and $iz = 1$ (default: 2).

tbeg, hmin, hmax: Beginning, minimum, and maximum time step sizes. $hmin$ is the minimum time step allowed. $hmax$ should be chosen to ensure accuracy and to save time. If $hmax$ is initially set to zero, it will be optimized based on beam energy. $ssdt$ must be greater than $hmax$ so all snapshot times accumulate charge density. See Section 6.10 (defaults: 1.e-11, 5.e-15, 0.0).

tbeg: Start time (default: 0).

tmax: Cutoff time for trajectories in seconds. This limit avoids excess time following tortuous trajectories (default: 1.e-05).

eps, freps, errmx: Error parameters for integrator. For definitions, see Section 5.5.2 (defaults: 5.e-04, 0.1, 10.0).

ssdt: Snapshot time interval for charge density calculations. At each snapshot time, charge density is allocated to the four corners of the surrounding grid cell. $ssdt$ should be chosen greater than the maximum allowed step size. See Section 6.10 (default: 1.5e-9).

xs(i): Starting x coordinate value for i th beam component. Eight different beam components are allowed in the present version of the DART code. The total number of components is determined by the $energy(i)$ array. See Fig. 4 for a description of this and the following input variables (default: 0 for $i = 1, 8$).

zs(i): Starting z coordinate value for i th beam component (default: 0 for $i = 1, 8$).

bmw(i): Beam width for *i*th beam component (default: 0.06 for *i* = 1, 3, and 0 for *i* = 4, 8).

angle(i): Starting angle for *i*th beam component. Angle is measured in radians from (x, y) plane (default: 0 for *i* = 1, 8).

energy(i): Beam energy for *i*th component. When the first value of *energy(i)* equal to zero is encountered, the code assumes all energy components have been specified, and the run is completed. Eight energies are allowed. To start a beamlet with zero energy, choose an energy that is negligibly small but nonzero (default: 100, 50, 33, and 0 for *i* = 4, 8).

ntraj(i): Total number of trajectories to calculate for the *i*th beam component. See Section 6.4 (default: 20 for *i* = 1, 3; 0 for *i* = 4, 8).

numpl: Maximum number of trajectories per plot. Choosing a smaller value of *numpl* will help save memory: plotting arrays are dynamically dimensioned with *numpl* (default: 20).

z0(i): Scale factor, *z*₀, in the Gaussian distribution for *i*th beam component:

$$f = f_0 \exp - \left(\frac{r - z_1}{z_0} \right)^2.$$

If *z0(i)* = 0, a flat-top distribution is used; ions are distributed uniformly across the beam with the same partial current: $I_p = \text{tcrrt}(i)/\text{ntraj}(i)$. See Section 6.4 for details (default: 0.0632 for *i* = 1, 3; 0 for *i* = 4, 8).

z1(i): *z*₁ for the *i*th beam component in the expression above. (default: 0 for *i* = 1, 8).

tcrrt(i): Total current density (in A/m) for each beam energy (*i*). If the beam is adjacent to a symmetry plane, enter only half the total current. For a description of use, see Section 6.4 (default: 9.7, 1.05, 0.455, and 0 for *i* = 4, 8).

3.4 Parameters Contained in Namelist /nl4/

wcold: Starting energy of cold ions (default: 1.e-04).

press: Gas pressure (in Torr) for cold ion production (default: 2.e-04).

nctr: Maximum number of cold ions to be followed. For description, see Section 6.9 (default: 10).

id: Identifier for cold ion cross-section calculations. Values of *id* correspond to different combinations of ions/background gas: *id* = 1 means He⁺/He; *id* = 2 means He⁺/H₂; *id* = 3 means H⁺/H₂, D⁺/D₂, or T⁺/T₂ (default: 3).

iprint: Number of times to skip between prints to output file. If *iprint* is nonzero, the code prints time, position, velocity, kinetic energy, and total energy. If 0, no print is made (default: 0).

isc: Switch to indicate continuation of space charge trajectory iterations. When *isc* = 0, DART starts with a vacuum potential calculation. When *isc* = 1, the potential table from a previous run is read from the file POTAB. An error will result if *isc* = 1 and POTAB does not exist (default: 0).

igcd: Switch to calculate charge density table when set to 1. When *igcd* = 1, QDTAB is used or is created. When *igcd* = 0, charge density is not accumulated (default: 1).

iqtab: Switch to accumulate charges (1) or create a new table (0) for charge densities. DART adds to QDTAB when *iqtab* = 1 and creates QDTAB when *iqtab* = 0. This switch is only used if *igcd* = 1 (default: 0).

ispar: Switch to include secondary electrons. If set to 1, trajectories originate from impact locations on electrodes. For more details, see Section 5.6 (default: 0).

icold: Switch to include cold ions. Cold ions are included when *icold* = 1 (default: 0).

icsc: Switch to calculate space charge contribution from cold ions. Cold ion space charge is included when *icsc* = 1 (default: 0).

itmax: Maximum number of space charge trajectory iterations (default: 1).

xlev: Spacing for potential contours in kV (default: 5).

sest: Scaling factor for secondary electron calculations. *sest* multiplies γ , the secondary electron emission coefficient. See Section 5.6 (default: 1.0).

pmp: Percentage of maximum production rate for cold ions as calculated by DART. *pmp* is used to determine whether a cold ion trajectory should originate from a given grid-point location based on the cold ion production rate calculated for that point (If $cipror(i,k) > amxpr \cdot pmp$, DART starts a cold ion trajectory, where *amxpr* is the maximum rate calculated for the entire grid and $cipror(i,k)$ is the production rate for the grid point.) See Section 6.9 (default: 0.1).

3.5 Parameters Contained in Namelist /nl5/

Electrodes (*numel*) can number between one and nine. A fixed boundary is input with the electrodes (boundaries are explained in Section 5.3). The potential of the *n*th electrode (or wall) is assigned with the array value, *vinput*(*n*). Electrodes are input with arrays *ix* and *kz*. All input data are contained in namelist /nl5/.

ix(*n*, *j*): Array containing the *x*-grid values for the *n*th electrode. The *j*th value is the *x* location of the *j*th point. The last entry must be “-1”, which signifies the end of the data string for this electrode. The code automatically reconnects the last point to the first, so for an electrode with four corners, *j* ranges from 1 to 5, with *ix*(*n*,5) = -1.

kz(*n*, *j*): Array containing the *z*-grid values for the *n*th electrode. The *j*th value is the *z* location of the *j*th point. The same rules apply as in the *ix* array. Remember that the last entry must be “-1”.

numel: The total number of electrodes (including walls). In the test case described in Section 4. we set *numel* = 5 (default: 1).

vinput(*n*): The potential assigned to the *n*th electrode (default: 0.0 for *n* = 1, 9).

We have tried to limit the first two namelist groups to variables that are least likely to be changed from one DART run to the next. The last three namelist groups are displayed to the terminal and the user is prompted for changes. We will now describe how to construct the input file, and how to make changes from the terminal.

4 Input and Output Files

4.1 Input File DARTIN

DARTIN is the input file needed to run DART. It contains each of the namelists mentioned in Section 3. The simplest input file will run the default case:

```
$endnl1

$endnl2

$endnl3

$endnl4

(ix(1,j),j = 1,7) = 0 0 5 5 2 2 - 1
(kz(1,j),j = 1,7) = 34 6 6 16 16 34 - 1
(ix(2,j),j = 1,5) = 7 7 16 16 - 1
(kz(2,j),j = 1,5) = 11 7 7 11 - 1
(ix(3,j),j = 1,11) = 20 20 23 29 29 45 45 31 31 24 - 1
(kz(3,j),j = 1,11) = 20 18 18 12 7 7 9 9 13 20 - 1
(ix(4,j),j = 1,5) = 0 60 60 60 - 1
(kz(4,j),j = 1,5) = 40 40 6 40 - 1
(ix(5,j),j = 1,3) = 0 0 - 1
(kz(5,j),j = 1,3) = 5 0 - 1
numel = 5
vinput = -100. -140. -10. 0. -100.
$endnl5
```

Statements with dollar signs (e.g., `$endnl3`) indicate the end of a namelist block (`/nl3/` in this case). The dollar sign says that you are through listing data for that namelist and are ready to begin listing data for the next namelist (in this case, `/nl4/`).

This default case is one space charge iteration with three different ion energies in the geometry designed for the neutral beams in the Tore-Supra experiment [3]. The graphics output generated by the DART code is shown in Figs. 5–7.

Default values for the electrode arrays are zero. You must specify electrodes in DARTIN.

After DART reads from DARTIN, it writes the namelists containing input parameters to the terminal. Then it writes,

Make changes to this namelist here (end with a \$):

You may then adjust any variable in the list to new values. Type them as they appear in the list, separating them with spaces. Carriage returns are ignored. When done, type “\$” and a carriage return. *If you don’t want to change anything, just type a “\$” and press return.*

For example, after the first namelist is displayed and you are prompted with the above message, you can type:

pref = 0. \$

If you want to add an additional electrode from the terminal, set *numel* to the new value (add 1) and put the last elements of *ix* and *kz* to -1, as described in Section 3.

If you entered data as specified but are unsure that DART is running, type “ctrlE s” and return. If you see

RCV + *darta* ,

DART is waiting for more input. Type “\$” and return.

4.2 Output File DARTOUT

DARTOUT is an output file generated by running the DART code. It is useful for diagnostic purposes and lists (1) the input variables for identifying the run and checking values, and (2) the arrays *nelec*, which are useful for checking that the electrodes were entered properly. The list of arrays consists of zeros where there are no electrodes, ones where electrode one should appear, twos for electrode two, and so on up to the maximum nine electrodes. DARTOUT also gives the potential calculations at various locations to check how many steps are needed for convergence of the relaxation scheme. After listing the scaled position arrays and the potential for vacuum as well as each space charge iteration, it gives a history of each particle including initial and final positions, velocities, and energies. It then lists summaries for secondary electrons and cold ions, if they were included in the run. Then it starts over, listing potential values for the next space charge calculation. DARTOUT will be very long if many space charge iterations are performed (i.e., if *itmax* is large), so you may not want to print the entire file.

DART also produces an FR80 graphics [2] file that contains input data, the *nelec* table, described in the DARTOUT description, contours of vacuum and each space charge iteration of the potential, and a plot of the trajectories. The computer-generated pictures in Figs. 5–7 are examples of frames from this file. This graphics file also displays summary tables of primary trajectories, secondary electron trajectories, and cold ion trajectories. At each stage the current collected at each electrode is given, as is the total current from all particles for each electrode.

The FR80 file always begins with “f3” and the first one will be called “f3dart0x”. The third character is incremented alphabetically as newer files are generated. The FR80 file can be printed at the default site (i.e., site=III) by typing,

netplot f3dart0x / t v

With DART input and output described, we are now ready to discuss the internal workings of the DART code itself.

5 What DART Does

This section describes the mechanics of the DART code. We begin by discussing the equations of motion; then we describe trajectory calculations with space charge. Finally, we describe the optional computation of secondary electron and cold ion trajectories.

Figure 5: Vacuum potential contours for a direct converter design as generated by DART with default values.

Figure 6: Space charge potential contours as generated by DART with default values.

Figure 7: Ion trajectories as generated by DART with default values. (a-c) Trajectories in the vacuum potential; (d-f) resulting trajectories in the first space charge iterated potential.

5.1 Solving the Equation of Motion

The second-order differential equation describing charged particle motion in an electromagnetic field,

$$q\vec{E} + q(\vec{v} \times \vec{B}) = m\vec{\ddot{r}}, \quad (1)$$

is decomposed into a set of first-order differential equations of the form,

$$\frac{d}{dt}Y_i = F_i. \quad (2)$$

By setting

$$\begin{aligned} Y_1 &= x & Y_4 &= \dot{x} \\ Y_2 &= y & Y_5 &= \dot{y} \\ Y_3 &= z & Y_6 &= \dot{z}, \end{aligned} \quad (3)$$

we obtain

$$\begin{aligned} F_1 &= Y_4 & F_4 &= \frac{q}{m}E_x + \frac{q}{m}(\dot{y}B_z - \dot{z}B_y) \\ F_2 &= Y_5 & F_5 &= \frac{q}{m}E_y + \frac{q}{m}(\dot{z}B_x - \dot{x}B_z) \\ F_3 &= Y_6 & F_6 &= \frac{q}{m}E_z + \frac{q}{m}(\dot{x}B_y - \dot{y}B_x). \end{aligned} \quad (4)$$

The Adams method used to solve these equations numerically is described briefly in Section 6.5.

5.1.1 Calculating Forces

The electric field that a particle encounters is given by

$$\vec{E} = -\nabla\phi. \quad (5)$$

By the principle of superposition, each component is treated independently. The potential ϕ is determined from Laplace's equation to get the vacuum potential and by Poisson's equation to get the potential with space charge.

The two-dimensional array *igecom* consists of ones and zeros. It determines the electrode locations and is also used in the potential calculation. The relaxation algorithm is used to calculate potential when *igecom*(*i*, *k*) = 0, and the potential at the (*i*, *k*) location is fixed when *igecom*(*i*, *k*) = 1. The relaxation scheme is described in Section 6.2.

If a boundary is indicated by maximum or minimum values of coordinate arrays and no wall is specified, the relaxation algorithm assumes reflection symmetry. DART accomplishes this simply by assigning values just outside the boundary to be equal to the values just inside; values on the boundary are calculated by relaxation. Boundaries are described further in Section 5.3.

5.2 Space Charge Calculations

Space charge effects are calculated in two steps. First, the charge density table is produced. Then, the method of relaxation is used to solve Poisson's equation and determine the new potential table. The number of trajectory iterations to be done in a given run is determined by the input parameter *itmax*. If further iterations are required to get a converged solution, you can run DART again with *isc* = 1. The code then reads the potential table generated in the last iteration from disk file POTAB, which was created during the previous run.

For details on charge density calculation, see Section 6.10, and for details on the potential calculations, see Sections 6.2 and 6.3.

Figure 8: Symmetry plane specifiers at corresponding planes. When the specifier is set to 1, its plane is a symmetry plane; when the specifier is set to 0, its plane is a transparent plane.

5.3 Boundaries: Transparent, Fixed, Symmetry

The DART code has three kinds of boundaries. To distinguish them, we call them fixed potential boundaries, symmetry planes, or transparent boundaries.

The default is a transparent boundary. Such a boundary represents the end of a device without a wall. Where particles should continue beyond the edge of the field, we actually continue to integrate the equations of motion and to follow a trajectory for one cell beyond the boundary, a procedure that allows charge density contributions to be calculated more precisely at the boundary. The plotted trajectory paths terminate at the boundary (as they do always), with the number of the trajectory displayed. The potential calculations are performed by use of symmetry. A fictitious row of values is set one cell length beyond the boundary. The values are set equal to potential values one cell length before the boundary, and the relaxation is performed up to and including the boundary.

A second type of boundary is the fixed potential boundary. By using electrode arrays ix and kz , we can specify walls at a fixed potential. Only two points in these arrays (plus the all important -1) are necessary. The potential boundary values do not change during relaxation.

If you continue a wall around a corner without returning to that corner, the resulting calculation will be indistinguishable from that for an electrode occupying the corner. If fixed potential boundaries are input with two points only (plus the -1), this situation is avoided.

The third boundary is the symmetry plane. The default incorporates a symmetry plane at $z = 0$. We use this boundary to save memory and potential calculation time by using the symmetry often found in such devices. To make a transparent boundary as on the other three walls, set $isyp(1) = 0$. You also may add a symmetry plane at $z = zmax$ by setting $isyp(2) = 1$. Symmetry planes at $z = 0$ and $z = zmax$ are constructed by setting $isyp(3) = 0$, or $isyp(4) = 0$, respectively. The symmetry plane arguments are depicted in Fig. 8.

The DART code contains a variable called $igrd$. This variable indicates which electrode was impacted by a given trajectory. $igrd$ is less than zero when a trajectory impacts a fixed boundary and is zero when a trajectory impacts a transparent boundary.

5.4 Starting Beams at Various Angles

The initial conditions for a beam component allow multiple beams from different starting positions and with different orientations. The beam components may also differ in energy and current profile. A beam component will start from within the following arrays:

- $xs(i)$ is the x coordinate of the starting point for the i th beam component.
- $zs(i)$ is the z coordinate of the starting point for the i th beam component.
- $angle(i)$ is the angle from the (x, z) plane of the i th beam component.
- $bmw(i)$ is the width of the i th beam component.

The energy and current are specified as follows:

- $energy(i)$ is the energy of the i th beam component.
- $z0(i)$ and $z1(i)$ are the current distribution parameters of the i th beam component.
- $tcrrt(i)$ is the total current (in A/m) of the i th beam component.

These input arrays are discussed in Section 3. The orientation for a given beam component is shown in Fig. 4. To start individual trajectories, set $ntraj(i) = 1$ and $bmw(i) = 0$.

5.5 Determining End Points and Efficiencies

5.5.1 Conditions for Terminating Trajectory Calculations

There are four conditions for terminating trajectory calculations:

1. If the value of the smallest allowable time step $hmin$ is not small enough to maintain restriction on the local relative error, which is bounded by eps .
2. If the time used in integration exceeds the input parameter $tmax$.
3. If the trajectory exits beyond the boundaries specified by $xmin$, $xmax$, $zmin$, or $zmax$. The output parameter is $igrid = 0$ in this case.
4. If the trajectory intersects an electrode (or a wall). Here $igrid$ is the number of the impacted electrode. (It is negative for a wall.)

The most frequent (and most desired) form of termination is the last condition. Details on how an impact is detected are given in Section 6.11.

5.5.2 Efficiency Calculations and Energy Conversion Check

DART calculates the efficiency of a given ion trajectory from its initial and final potential energies. The efficiency is given by

$$\eta = \frac{\phi_f - \phi_0}{W_i + \phi_i - \phi_0}, \quad (6)$$

where ϕ_i and ϕ_f are the initial and final potential energies in kV; W_i is the initial kinetic energy in keV; and ϕ_0 is the reference potential input as *uref*, also in kV.

The relative error in the total ion energy is given by the formula

$$W_{err} = \left| \frac{(W_f + \phi_f) - (W_i + \phi_i)}{W_i + \phi_i} \right|, \quad (7)$$

where the final kinetic energy, W_f , is calculated from the final velocity components. If W_{err} is greater than *errmx*, the relative error test constant for the integrator, *eps*, is adjusted by multiplying it times the fraction, *freps*. Then the trajectory is recalculated.

5.6 Secondary Electrons

Secondary electrons are included in a DART run if the switch *ispar* = 1, and are not included if *ispar* = 0. In subroutine SECPAR, secondary electron trajectories are started where an ion impacts an electrode, and they are initially directed back along the path of the incoming ion. Each trajectory carries a current equal to γ times the current of the ion trajectory, where γ is the secondary electron emission coefficient and is a function of impact energy. γ is calculated in subroutine GAMMA; *sesf* is a multiplier for γ to adjust for different materials.

The true secondary electron emission coefficient is actually a function of many variables: charge and mass of the incident ion, target material (e.g., atomic number, atomic weight, electron structure), angle of incidence (and therefore the roughness of the electrode surface), and especially the chemical composition of the surface layer (e.g., gas layer, oxide layer). DART uses a simple expression that is a reasonable fit to the data for D^- on clean Mo and that also fits the data for D^+ on W and He^+ on Mo and W. In Fig. 9, DART's expression is superimposed on L.S. Hall's Fig. 4 [4]. The high values are for gassy or oxide surfaces. The emission is thought to vary as the cosecant of the angle between the incident ion and the normal vector to the surface. We did not include this angle dependence because we did not have a reference that describes this effect quantitatively.

The formula used in subroutine GAMMA is

$$\gamma \equiv sesf \begin{cases} 0.1 & \text{if } W \leq 0.1 \text{ keV} \\ 0.3\sqrt{W} & \text{if } 0.1 \leq W \leq 100 \text{ keV} \\ 3.0 & \text{if } W \geq 100 \text{ keV}, \end{cases} \quad (8)$$

where the proportionality factor, *sesf*, is an input parameter.

5.7 Cold Ions

An ion beam produces cold ions and electrons along its path by ionizing gas molecules, by dissociating gas molecules, and by charge exchange with gas molecules. The cold electrons contribute to the Maxwellian distribution of electrons that DART includes in the calculation of space charge. The cold ions not only contribute to space charge, but also deposit electrical current and heat on the electrodes where they are collected. The cross sections used by DART for determining the production rate of cold ions are shown in Figs. 10 and 11, both taken from C.F. Barnett et al. [5]. The user is cautioned that DART's cross sections may be inaccurate for H^+ energies below 0.5 keV or above 2 MeV.

Cold ions can be started by setting, *icold* = 1. No cold ion trajectories are calculated if *icold* = 0. The array *cipror* is filled with values in subroutine QDENT, and cold ion trajectories are calculated in subroutine CIPROD. Details are described in Sections 6.10 and 6.9. The cold ion energy is given by the input parameter called *wcold*. The background gas pressure is *press* and the cross-section identifier is *id* (see Section 6.9).

Section 6.9 describes how cold ion trajectories are calculated, and Section 10 gives a simple test example for cold ions.

- (a) He⁺ on Mo
- (b) He⁺ on (111) face of Ni
- (c) H⁺ on Mo
- (d1) H⁺ on Cu; (d2) H⁺ on Mo and 302 stainless steel
- (e) He⁺ on Mo
- (f1) He⁺ on Mo; (f2) H⁺ on Mo
- (g1) H⁺ on Mo; (g2) D⁺ on Mo; (g3) H⁺ on Cu
- (h1) He⁺ on W; (h2) H⁺ on W
- (i1) H⁺ on Cu; (i2) H⁺ on Cu; (i3) H⁺ on Mo;
- (j1) H⁺ on C; (j2) H⁺ on Pt; (j3) H⁺ on Mo; (j4) H⁺ on Cu; (j5) H⁺ on Ti
- (k) H⁺ on stainless steel
- (l1) H⁺ or D⁺ on C; (l2) D⁺ on Pt; (l3) He⁺ on C; (l4) He⁺ on Pt

Figure 9: Secondary electron emission coefficient, γ , is shown as calculated in DART, superimposed on Hall's data. The table above is also from Reference [4].

Figure 10: Cross section used to calculate production of positive ions by He^+ ions in H_2 and He . DART uses the values shown by the dashed line. Data are from Reference [5], Fig. A.5.62.

Figure 11: Dashed line shows total cross section used by DART to calculate the production of cold ions and electrons by H^+ in H_2 . Curves (a) and (b) are from Reference [7], Fig. A.4.5 (charge exchange) and Fig. A.5.7 (ionization by protons). For other isotopes of hydrogen, DART adjusts the energy scale to give the same velocity as H^+ .

6 Major DART Subroutines

Each subroutine and related physics are described in the subsection bearing its name. For the most part subroutines are listed in the order in which DART's main subroutine calls them (Figs. 12, 13).

6.1 Designating Electrodes

6.1.1 GETGRID

The subroutine GETGRID calls the more specialized subroutine ELECTROD to set *geom*, an array of ones and zeros indicating electrodes and spaces, respectively. *input(n)* is used to assign the potential to the *n*th electrode region. GETGRID is called only once, when the code is initialized.

GETGRID calls ELECTROD *numel* times, where *numel* is the number of electrodes. Each electrode is then added to the array *geom*. If any two electrodes overlap, an error message is printed to the terminal, and the code exits. First, however, *geom* is printed to the output file, DARTOUT. To find the input error, look at DARTOUT with *tedi* or another editor, or type "type dartout". Overlapping electrodes are indicated by twos in the *geom* array. Electrodes are allowed to lie next to one another, but they cannot share points.

6.1.2 ELECTROD

When the difference in consecutive *z*-grid coordinates from the array *kz(n,j)* is negative, values to the right of the corresponding *ik(n,j)* entries and the line interpolated between the *j*th and (*j* + 1)th points have a +1 added. When the *z* difference is positive, values to the right are changed by -1. This basic principle is used to construct the electrodes. If the direction changes (left or right) while consistently moving down (or up), adding (or subtracting) twice is avoided.

Since the diagonal surface of an electrode will have the constant potential value of that electrode, inside corners only exist to the grid-scale size. In other words, consider a block of four grid points. An inside corner is represented by three ones and a zero. A diagonal boundary on an electrode is also represented by this configuration. Since there is no difference, we decided to treat either case as a diagonal boundary. This decision eliminates the existence of true inside corners on electrodes, and is observable in the DART-generated figures. For example, the upper right-hand corner of our default test case contains a small triangle. This triangle indicates that the wall input as electrode 4 bends from across the top down to the right side (see Fig. 5).

6.2 POT1

POT1 calculates the vacuum potentials for the geometry calculated by GETGRID. It uses the relaxation algorithm, where no relaxation is done on the electrodes (i.e., where *geom(i,k)* = 1).

POT1 solves Laplace's equation in two dimensions

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = 0, \quad (9)$$

and the relaxation formula is derived by using the central differences

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi_{i+1,k} - 2\phi_{i,k} + \phi_{i-1,k}}{(\Delta x)^2} \quad (10)$$

$$\frac{\partial^2 \phi}{\partial z^2} = \frac{\phi_{i,k+1} - 2\phi_{i,k} + \phi_{i,k-1}}{(\Delta z)^2} \quad (11)$$

Figure 12: Calling sequence and control for major DART subroutines. The main subroutine (a) calls the initializing subroutine in the order shown, then (b) calls four subroutines in a loop until all trajectory iterations are completed.

Figure 13: DART subroutine DRIVESA and function PHI call their own subroutines.

Substituting these into Eq. 9 gives the relaxation formula:

$$\phi_{i,k} = \frac{\phi_{i+1,k} + \phi_{i-1,k} + \left(\frac{\Delta x}{\Delta z}\right)^2(\phi_{i,k+1} + \phi_{i,k-1})}{2[1 + \left(\frac{\Delta x}{\Delta z}\right)^2]}. \quad (12)$$

Relaxation involves iterating Eq. 12. The rate of convergence is increased by over-relaxation as defined by

$$\phi_{i,k}^n = \phi_{i,k}^{n-1} + \alpha(\phi_{i,k} - \phi_{i,k}^{n-1}), \quad (13)$$

where $\phi_{i,k}$ is the result from Eq. 12, n is the iteration number, and $\alpha > 1$ (typically $\alpha \approx 1$). The file DARTOUT contains displayed selected values of ϕ at various points to check convergence.

6.3 POT

The potential subroutine is called POT. On its first call, POT calls GETGRID to set up where the electrodes are located. If $isc = 0$, it calls POT1 to get the vacuum potential with this configuration. POT uses a relaxation technique to calculate space charge potentials due to charge densities calculated in QDENT and the electron distribution described in Section 7.

To calculate the space charge potential, we must solve Poisson's equation,

$$\nabla^2 \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = -\frac{\rho}{\epsilon_0}. \quad (14)$$

The relaxation algorithm derivation is analogous to the derivation shown in the last section and is given by

$$\phi_{i,k} = \frac{\left(\frac{\Delta x}{\Delta z}\right)^2 g_{i,k} + \phi_{i+1,k} + \phi_{i-1,k} + \left(\frac{\Delta x}{\Delta z}\right)^2(\phi_{i,k+1} + \phi_{i,k-1})}{2[1 + \left(\frac{\Delta x}{\Delta z}\right)^2]}, \quad (15)$$

where ϵ_0 is the dielectric constant of free space and $g_{i,k}$ is the net charge density. $g_{i,k}$ is set to zero if the electron contribution should outweigh the positive ion contribution.

Since the charge densities are constantly changing with time, a summation of charge density laid down by each trajectory is used. A description of the snapshot technique used in the DART code is found in Section 6.10.

6.4 START

START generates the starting conditions for DART. It initializes particle positions, energies, and currents. The currents may be distributed among trajectories with a Gaussian profile. If $z0(i) = 0$ for the i th set of particles, a uniform distribution is used instead.

The distribution function for a Gaussian profile is

$$f(z) = \exp \left[- \left(\frac{z - z_1}{z_0} \right)^2 \right], \quad (16)$$

where z_0 and z_1 are input in the arrays $z0(i)$ and $z1(i)$, with the index i corresponding to a particular beam component.

DART calculates the current density by dividing the total current (for the half geometry) given by $icrnt(i)$ by the area under the curve of $f(z)$ over the beam width, $bmw(i)$. The current assigned to a given trajectory is this current density times $f(z)$ times the spacing between initial positions of the particles. The spacing is determined by the input parameters, $ntraj(i)$ and $bmw(i)$ (see Fig. 4).

6.5 The ODE Solver

The ODE solver found in DART is a predecessor of LSODE (the Livermore Solver for Ordinary Differential Equations) [6]. Both were written at LLNL by A. Hindmarsh and use the Adams method for solving a system of first-order ordinary differential equations. The subroutines used by DART are DRIVESA, STIFFA, COSETA, and FUNCT, all documented by comment statements in the *dart!* listing. A brief description of what they do and how they were adapted for DART follows.

6.5.1 DRIVESA, STIFFA, COSETA, and FUNCT

DRIVESA is the driver subroutine for STIFFA. Within DRIVESA, after STIFFA has been called, plotting data are stored, subroutine QDENT is called to generate charge density tables, subroutine IMP is called to check for impacts, reflection at symmetry planes is accomplished (when *isym*(*i*) = 1), time is checked, and appropriate error messages are printed when problems are encountered.

STIFFA does one step of the integration and uses COSETA to obtain coefficients. It also optimizes the step size.

FUNCT contains the functions that determine the derivative of the integration variables. In other words, it gives $F(Y, t)$ in the equation

$$\frac{dY}{dt} = F(Y, t),$$

where Y is the vector valued function given in Eq. 3. In DART, this is where velocities and forces are calculated from the electromagnetic fields. $F(Y, t)$ is given in Eq. 4.

6.6 PHI and LAG2

PHI is the subroutine that gets the potential at a given location in x and z . PHI calls on LAG2 to interpolate the table established by POT. LAG2 is a standard interpolator, and the degree of the polynomial used for interpolation is set with the *parameter* *ne* in the source file, *dart!*. The present version of DART is set to interpolate with second-degree polynomials. Again, to change this requires editing the source file. When a boundary is encountered, the interpolator uses the last *ne* + 1 points up to the boundary.

6.7 PARPLT

PARPLT produces the picture of the particle trajectories superimposed on the electrode configuration. The example shown in Fig. 7 was produced by DART with the default input parameters. The potential value of a given electrode is printed one grid-point down and to the right of the upper left-hand corner of an electrode. Cold ion trajectories are printed with dashed lines, and secondary electron trajectories are printed with dotted lines.

6.8 SECPAR

SECPAR is the secondary electron subroutine and is called if the switch *ispar* = 1. In subroutine SECPAR, a secondary electron trajectory is started where an ion impacts an electrode. The trajectory is initially directed back along the path of the incoming ion. Each trajectory carries a current equal to γ times the current of the ion trajectory, where γ is the secondary electron emission coefficient. A detailed description of γ is given in Section 5.6. SECPAR is called for each beam

component separately. If $icold = 1$, secondary electron trajectories are calculated from cold ion impacts when SECPAR is called from CIPROD.

6.9 CIPROD

CIPROD uses the production rate table called *cipror* to determine starting conditions for cold ion trajectories. It is called only when $icold = 1$. The maximum number of cold ion trajectories to be followed is determined by *nctr*. When an element of *cipror* is greater than the input parameter, *crpror*, a trajectory is set to start at that location. If the number of starting points exceeds *nctr*, then every other starting location is used. If there are still more starting locations than *nctr*, then every third location is used, and so on, until the number is less than *nctr*. The current assigned to each cold ion trajectory is given by the number of positions skipped plus one, multiplied by the area of a grid cell and by the cold ion production rate.

The production rate table is calculated in subroutine QDENT simultaneously with the charge density table. Then the ion trajectory velocity and the charge density can be used directly. The production rate formula is

$$cipror(i,k) = n_0 \sigma v \rho, \quad (17)$$

where n_0 is the background density determined from the gas pressure input parameter, *press*, and the conversion factor, $3.5 \times 10^{22}/\text{Torr}^{-1} \text{ m}^{-3}$; σ is the cross section for cold ions calculated in subroutine SIGMA; v is the velocity of beam ion trajectories; and ρ is the charge density.

SIGMA is the subroutine containing ionization and charge exchange cross sections for cold ion production. These cross sections depend on the local velocity of beam ions and on the ion and gas species. The identifier, *id*, is an input parameter and assumes the following values:

- $id = 1$: He^+ ions in He
- $id = 2$: He^+ ions in H_2
- $id = 3$: H^+ ions in H_2 , D^+ ions in D_2 , or T^+ ions in T_2

For default cases, no space charge is accumulated from cold ions. If cold ion space charge is important, set *icsc* = 1. Then QDENT will be called for cold ions as well.

6.10 QDENT

QDENT generates the charge density table used by POT. After a snapshot interval *ssdt*, the previous charge density variables are linearly interpolated to determine a new one. Charge is then distributed along the mesh points in a scheme devised by C.K. Birdsall [7]. The following description is taken from the original DART document [1]. References in that document indicate sources of further information.

In a given interval of time, Δt , the charge density contribution of a given particle is

$$q_d = I_j \Delta t. \quad (18)$$

Δt is the "snapshot" time. If all ions at a given energy are launched at the same time, then at a time Δt later, the charge densities are recorded, and again at a time Δt later, etc. These pictures may be superimposed to get a table of charge densities. For this to work, the number of particles must be great and the snapshot time must be small enough to get a smooth density distribution. To keep Δt small without changing the step size of the integration, DART linearly interpolates

between points of a trajectory. The particle location is calculated by interpolation and the charge density is distributed to the four surrounding mesh points to construct the charge density table.

The Birdsall scheme is given by

$$\begin{aligned}\rho_{i,k} &= \rho[(x_{i+1} - x)(z_{k+1} - z)/(\Delta x \Delta z)] \\ \rho_{i+1,k} &= \rho[(x - x_i)(z_{k+1} - z)/(\Delta x \Delta z)] \\ \rho_{i+1,k+1} &= \rho[(x - x_i)(z - z_k)/(\Delta x \Delta z)] \\ \rho_{i,k+1} &= \rho[(x_{i+1} - x)(z - z_k)/(\Delta x \Delta z)],\end{aligned}\tag{19}$$

where

$$\rho = \frac{q_d}{(\Delta x \Delta z)}.\tag{20}$$

In Eq. 20, q_d is the charge density (C/m) contributed by the particle to the table at each multiple of Δt .

The table is cumulative for each energy run specified by the *energy(i)* array at input. The final table is saved in the disk file QDTAB. When further accumulation of charges is desired (for example, if another beam were added), run DART again with *iqtab* = 1.

Remember that snapshot time must be short enough to record charge density a few times per cell based on the maximum velocity. The actual charge density is independent of snapshot time. When the snapshot time is reduced, more snapshots occur per cell and, to compensate, the charge distributed is less per snapshot.

Also, the snapshot time should be greater than the maximum step size allowed, *hmax*, to ensure that all snapshot times in a trajectory have charge distributed. QDENT is called when *t* exceeds *sst* + *ssdt*, where *sst* starts at one time step after the start time, *tbeg*. Then *sst* is incremented in intervals of *ssdt*, the snapshot time. QDENT interpolates position to the location at time *sst* + *ssdt*, and distributes charge from this location. If *hmax* is too large, the timesteps may exceed the snapshot time. Then interpolations will be done backward, and the last snapshots may not take place. If *ssdt* is less than *hmax*, a warning message is printed to the terminal.

Geometrically the Birdsall scheme represents weighting the charge density allocated to a corner of a grid cell by the fraction of the total area contained in the opposing quadrant (see Fig. 14).

Weighting charge density for a symmetry plane is handled as follows. In a cell that has the symmetry plane as a border, all trajectories contribute twice the normal charge to the points in the symmetry plane. This is due to the neighboring trajectories that are supposed to exist on the other side of this plane. A trajectory along this plane will contribute only its normal share, allowing a beam traveling parallel to the symmetry plane to have one trajectory contained within this plane if desired. Having a trajectory in the symmetry plane is typically avoided by the choice of initial conditions. The width of the beam is divided into *ntraj(i)* equal columns. One trajectory is then started down each column (see Fig. 4).

A beam that originates on a boundary has charge density at that boundary set equal to the charge density one grid row inward. This simulates a beam that originated exterior to our field. When constituents of the beam exit the working field (i.e., when trajectories leave the charge density table), we allow them to continue one grid row beyond to get a more accurate charge density contribution at the edge of the table. To start a beam at a fixed potential, we must start trajectories on a fictitious wall or electrode.

In addition to charge density calculations, the cold ion production rate table, *cipror*, described in Section 6.9, is calculated in subroutine QDENT.

<u>Grid Point:</u>	<u>Weighted Area:</u>
1
2
3
4

Figure 14: Bird'sall's scheme for charge density distribution. The fraction of charge from a snapshot allocated to a corner is determined by the ratio of area in the opposite rectangle to total cell area.

6.11 IMP

IMP is the impact subroutine. It determines whether the given particle trajectory has intersected an electrode. If so, it returns to DRIVESA, which terminates the trajectory and records final data.

IMP uses the array called *geom*, which consists of ones and zeros, calculated in GETGRID. This array designates the electrodes by values of 1, interspaced by values of 0. Then boundaries can be comprised of the diagonals across a given grid cell. A cell with corner values that consist of three ones and a zero is always considered to be a diagonal boundary. Thus, there is no such thing as an inside corner, which avoids confusion.

IMP checks the four corners of a given grid cell for electrode boundaries. If the sum for the four corners is less than three, the given particle has not impacted an electrode.

If the sum is four, a check is performed to determine whether all four points belong to the same electrode. If they do, an impact will occur. The code now interpolates back along the trajectory path, reduces the step size by a factor of ten, and resumes the trajectory calculation. When the electrode is encountered, the code again backs up along the trajectory and reduces the step size. At the third encounter, the position, time, and velocity are saved. If the four points do not belong to the same electrode, no impact is recorded and the trajectory calculations continue.

If the sum of the corner values is three, the boundary is diagonal. The code checks to see that all ones belong to the same electrode. If not, there is no impact. If they do, an impact occurs only if the trajectory is in the interior of the triangle with the ones as vertices.

If an impact occurs, the number of the electrode is recorded and the current is adjusted for that electrode. The array called *nelec* identifies the electrode. This array consists of the numbers zero through *numel*, and is displayed in DARTOUT and in the graphics output file.

If we have a hit, we record the number of the electrode and adjust the collected current on that electrode appropriately. The electrode is identified by checking the array called *nelec*. This array consists of the numbers zero through *numel*, and is displayed in DARTOUT and in the graphics output file.

7 Modeling Electrons

When space charge calculations are made and only ions are considered, often a potential builds up that is more positive than a nearby electrode. Such a buildup is nonphysical since the more mobile electrons will prohibit its existence. To model the electron behavior, DART uses two methods chosen by setting the switch called *ielmod*. No electron calculations are done when *ielmod* = 0. When *ielmod* = 1, we use a Boltzmann distribution of the form:

$$q_e = A \exp(\phi - V_g) \frac{1}{T_e}, \quad (21)$$

where V_g is the reference potential, T_e is the electron temperature, and A is given by

$$A = \mu \bar{q}_i. \quad (22)$$

$\bar{q}_i$ is the average ion charge density taken over a reference area. μ is e^{-3} and was introduced in this manner apparently to avoid subtracting 3 from the exponent argument repetitively. This factor represents the sheath region around an electrode by dropping the potential by a factor of $3T_e$. The electron temperature, T_e , is the input parameter, *te*.

If *ielmod* = 2 we use the reference potentials (plus $3T_e$) as upper bounds. If any calculated potential value in a given region is larger, it is set to the upper bound.

This electron modeling is done in two regions: region one is the entry region near the first suppressor in default geometry, and region two is the collector region in default geometry. The boundaries of region one are designated by the dashed line in Fig. 2. In this region, the area for calculating the average ion charge density, $\bar{q}_i$, is designated by the input parameters $il1$ and $kl1$, the lower limits, and $iu1$ and $ku1$, the upper limits for the x and z grid values in the area calculation. This area is enclosed by the dotted line in Fig. 2 and by the horizontally shaded area at the bottom of Fig. 3. Again, when $ielmod = 1$, the electrons are modeled by Eq. 21 with reference potential $V_g = vg1$, but only in the region where $inel < i < ipe1$ in x and $knel < k < kpe1$ in z (region one).

The arguments for electron modeling in region two are analogous. $vg2$ is the reference potential. The average ion charge density is calculated in the region bounded by $il2$ and $iu2$ in x and by $kl2$ and $ku2$ in z . The electrons modeled are referenced to the $vg2$ potential in region two (in the default case, this is everywhere outside the entry region—region one). Note that region one may be contained inside region two. However, region two may not be contained within region one. Region one always overrides region two when they overlap.

The electrons are incorporated into the relaxation loop in the code in an under-relaxation manner. Within the relaxation loop is the equation

$$q_{00} = q_{00} + 0.01 \left\{ A \exp(\phi - V_g) \frac{1}{T_e} - q_{00} \right\}, \quad (23)$$

where q_{00} is the electron contribution to the charge density. The total charge density is set to zero when the electron charge density exceeds the ion charge density. By adding only a small amount of the Boltzmann distribution at a time, oscillations in the charge density are avoided. Such oscillations are the result of relaxing to a potential value, altering the electron distribution, relaxing to a new potential value, etc.

8 A Beam with Space Charge—Test Case

As a test case for the DART code, we used a beam (originally uniform) traveling from the left to the right. Space charge calculations cause the beam to diverge. The top wall is fixed at 0.0 kV potential. The current and lengths are chosen to allow a one-dimensional treatment with velocity assumed uniform in x and constant electric field vertical (in z), which gives a constant acceleration in this direction. The beam thus has a quadratic profile, as shown in Fig. 15.

The input file used to generate these pictures is

```
imin = 0 imax = 50 kmin = 0 kmax = 31
xmax = 1. zmax = .31
vref = 0.
$endnl1

ielmod = 0
$endnl2

bmw = 0.05 0. 0. 0. 0. 0. 0. 0.
energy = 100. 0. 0. 0. 0. 0. 0. 0.
```

Figure 15: Beam test case showing vacuum potential (a) contours and (b) trajectories, and space charge potential (c) contours and (d) trajectories.

```

ntraj = 31 0 0 0 0 0 0 0
numpl = 31
z0 = 0. 0. 0. 0. 0. 0. 0. 0.
tcrnt = 0.548 0. 0. 0. 0. 0. 0. 0.
$endnl3

nctr = 10
zlev = 1.
$endnl4

(ix(1,j),j = 1,5) = 0 0 50 50 - 1
(kz(1,j),j = 1,5) = 31 30 30 31 - 1
numel = 1
vinput = 0.
$endnl5

```

The current is also chosen so the 5-cm beam will double in size to 10 cm after traveling 1 m. This test case was instrumental in improving methods for determining accurate charge density values on the midplane and across the beam to ensure their proper representation.

9 The Child-Langmuir Law—Test Case

The Child-Langmuir law is given by the equation

$$J = \frac{4}{9} \epsilon_0 \sqrt{\frac{2e}{m}} \frac{V^{3/2}}{x^2}, \quad (24)$$

where J is the current density (A/m), V is the potential at the object plate (V), x is the distance to the plate, $\frac{e}{m}$ is the charge to mass ratio for a species of charge particle, and ϵ_0 is the free space dielectric constant.

With the DARTIN file given here, the ions produce a space charge potential in agreement with Eq. 24:

```
imin = 0 imax = 41 kmin = 0 kmax = 20
xmax = 1.025 zmax = .5
vref = 0. isymp = 1 1 0 0
$endnl1
```

```
eps1 = 0.005 iternm = 800
ielmod = 0
ichi = 1
$endnl2
```

```
iz = 1 na = 4
bmw = 0.5 0. 0. 0. 0. 0. 0. 0. 0.
energy = 90.05 0. 0. 0. 0. 0. 0. 0.
ntraj = 40 0 0 0 0 0 0 0 0
numpl = 40
z0 = 0. 0. 0. 0. 0. 0. 0. 0.
ternt = .3675 0. 0. 0. 0. 0. 0. 0.
$endnl3
```

```
iprint = 1
$endnl4
```

```
(ix(1,j),j = 1,5) = 40 40 41 41 - 1
(kz(1,j),j = 1,5) = 20 0 0 20 - 1
(ix(2,j),j = 1,3) = 0 0 - 1
(kz(2,j),j = 1,3) = 0 20 - 1
numel = 2
vinput = 90. 0.
$endnl5
```

Figure 16: Child-Langmuir law test results. (a) The potential vs z distance for zero space charge iterations, and (b) potential contours, output potential values at increments of Δz from the right-hand side, and the E field at the left-hand side ($z = 0$). (c,d) same as (a,b), respectively, after three space charge iterations.

Figure 17: DART output for initial 100-keV ion trajectories.

This input file was used to generate the results shown in Fig. 16. For this test, we chose a uniform beam of 40 trajectories originating at 0 V and impacting an electrode with 90 kV. The initial energy of the beam is 90.05 keV, and the initial current is 0.3675 A/m. The potential as a function of distance is plotted with the solid line representing the DART result for the given number of iterations and the dotted line the exact analytic result.

Note The switch *ichl* should *only* be set to 1 for the Child-Langmuir law test because special graphs are generated (shown in Fig. 16) that are particular to this test case.

10 A Test Case for Cold Ion Calculations

Here we describe a simple test case to check the cold ion production calculations. This test consists of two electrodes with potentials of ± 100 V. A 4-cm beam of 100-keV He^+ ions passes between the electrodes from left to right through a background hydrogen gas. The generated 0.1-eV cold ions then fall to the negatively charged lower plate, where their current is collected.

The input file is as follows:

```
imin = 0 imax = 10 kmin = 0 kmax = 32
xmax = 1. xmax = .32
vref = 0. isymp = 0 0 0 0
$endl1

ielmod = 0
$endl2

zs = .14 0. 0. 0. 0. 0. 0. 0. 0.
bmw = .04 0. 0. 0. 0. 0. 0. 0. 0.
energy = 100. 0. 0. 0. 0. 0. 0. 0. 0.
```

summary of cold ions						
no.	x	y	z	st	igrid	current(A/m)
1	0.0000	0.0000	0.1400	1	-2	0.931044e-02
2	0.2000	0.0000	0.1400	1	-2	0.940336e-02
3	0.4000	0.0000	0.1400	1	-2	0.97626e-02
4	0.6000	0.0000	0.1400	1	-2	0.103970e-01
5	0.8000	0.0000	0.1400	1	-2	0.112620e-01
6	1.0000	0.0000	0.1400	1	0	0.121853e-01
7	0.1000	0.0000	0.1500	1	-2	0.105405e-01
8	0.3000	0.0000	0.1500	1	-2	0.105524e-01
9	0.5000	0.0000	0.1500	1	-2	0.105401e-01
10	0.7000	0.0000	0.1500	1	-2	0.105510e-01
11	0.9000	0.0000	0.1500	1	-2	0.105500e-01
12	0.0000	0.0000	0.1600	1	-2	0.105485e-01
13	0.2000	0.0000	0.1600	1	-2	0.105404e-01
14	0.4000	0.0000	0.1600	1	-2	0.105500e-01
15	0.6000	0.0000	0.1600	1	-2	0.105514e-01
16	0.8000	0.0000	0.1600	1	-2	0.105491e-01
17	1.0000	0.0000	0.1600	1	0	0.105535e-01
18	0.1000	0.0000	0.1700	1	-2	0.105485e-01
19	0.3000	0.0000	0.1700	1	-2	0.105524e-01
20	0.5000	0.0000	0.1700	1	-2	0.105485e-01
21	0.7000	0.0000	0.1700	1	-2	0.105516e-01
22	0.9000	0.0000	0.1700	1	-2	0.104943e-01
23	0.0000	0.0000	0.1800	1	-2	0.923001e-02
24	0.2000	0.0000	0.1800	1	-2	0.914401e-02
25	0.4000	0.0000	0.1800	1	-2	0.877304e-02
26	0.6000	0.0000	0.1800	1	-2	0.815201e-02
27	0.8000	0.0000	0.1800	1	-2	0.728365e-02
28	1.0000	0.0000	0.1800	1	0	0.636434e-02

total = 0.488041e+00

collected cold ion current(A/m) on electrode # 1 = 0.

collected cold ion current(A/m) on electrode # 2 = 0.370937e+00

Total collected current(A/m) for this beam component on electrode # 1 = 0.

Total collected current(A/m) for this beam component on electrode # 2 = 0.370937e+00

Cold ions were calculated skipping every 1 elements of the production rate table.

Figure 18: DART output for cold ion trajectories.

```

ntraj = 16 0 0 0 0 0 0 0
z0 = 0. 0. 0. 0. 0. 0. 9. 0.
tcrnt = 1.0 0. 0. 0. 0. 0. 0. 0.
$endnl3

nctr = 50
icold = 1
itmax = 0 xlev = .05
$endnl4

(ix(1,j),j = 1,5) = 0 10 10 0 - 1
(kz(1,j),j = 1,5) = 31 31 32 32 - 1
(ix(2,j),j = 1,5) = 0 10 10 0 - 1
(kz(2,j),j = 1,5) = 0 0 1 1 - 1
numel = 2
vinput = .1 - .1
$endnl5

```

The electrodes are placed 30 cm apart, and the beam has a total current of 1.0 A/m. The background gas pressure is 2×10^{-4} Torr, which gives a density of $n_0 = 7 \times 10^{18} \text{ m}^{-3}$. These values yield a cold ion production rate of 9.275 A/m³ (see Section 6.9).

The current collected at the lower electrode should then be 0.371 A/m. The results are shown in Figs. 17 and 18. The cold ion energy must generate a velocity. We chose this velocity in the positive x direction, which creates a slight drift of cold ion trajectories to the right.

11 Default Geometry

The geometry used in the neutral beams for the Tore-Supra tokamak experiment is briefly described in this section. This geometry was chosen as a default to demonstrate a good configuration with a sample of input data and typical design considerations. We also used it as a test to reproduce results obtained by J. Pamèla [3]. The vacuum potential is shown in Fig. 5, the potential with space charge in Fig. 6, and trajectories in Fig. 7.

The electrodes in Fig. 1 are as follows:

1. Part of the neutralizer apparatus. The beam enters at -100 kV.
2. Electron suppressor electrode. Its potential is -140 kV.
3. Ion recovery electrode. Its potential is -10 kV.
4. Walls of the energy recovery system. They are grounded at 0 kV.
5. Starting potential for the beam. This is -100 kV.

Figure 2 shows the regions used for electron modeling. In region one, the Boltzmann distribution is referenced to -100 kV. Elsewhere (region two), we reference to ground.

Potential contours for this default geometry are displayed in Figs. 5 and 6, and the ion trajectories subject to these potentials are shown in Fig. 7.

To model this system correctly, we can iterate space charge effects to convergence, include ionization and charge exchange due to cold ions, and include secondary electrons. We are currently performing more detailed studies with all of these considerations.

12 Summary

The present (1988) version of DART is nonrelativistic and does not include magnetic field input data. We leave relativistic alterations and magnetic field data input for future work.

The greatest asset of the present version is its geometry independence. An arbitrary electrode configuration with arbitrary starting conditions is easily represented by this code. We believe this new DART will be a useful tool for many applications.

Acknowledgments

We thank J. Paméla for supplying the geometry for the default test case, and Peter Willmann for his help with memory and timing studies.

The software for DART (DARTL) is available from the National Energy Software Center, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439.

References

- [1] B.C. Howard, W.L. Barr, and R.W. Moir, *DART: A Simulation Code for a Direct Energy Converter for Fusion Reactors*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-51557 (1974).
- [2] P.A. Willmann, *Graft III: User-Oriented Science Graphics*, Lawrence Livermore National Laboratory, Livermore, CA, M-208 (1987).
- [3] J. Paméla, "Preliminary Results on the Design Study of the Energy Recovery Prototype System for the Tore-Supra Neutral Injectors," *Note Technique TS 80-8604*, Caderache, France (1986).
- [4] L.S. Hall, *Nuc. Fusion* **17**, 681 (1977).
- [5] C.F. Barnett et al., *Atomic Data for Controlled Fusion Research*, vol. I, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-5206 (1977).
- [6] A.C. Hindmarsh, in *Scientific Computing*, R.S. Stepleman, Ed. (North Holland, Amsterdam, 1983).
- [7] C.K. Birdsall and D. Fuss, *J. Comp. Phys.* **3**, 494 (1969).